A set of parallel, implicit methods for a reconstructed discontinuous Galerkin method for compressible flows on 3D hybrid grids

Yidong Xia, Hong Luo, Megan Frisbey, Robert Nourgaliev

1 Department of Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, NC 27695, USA
2 B Division, Weapons and Complex Integration, Lawrence Livermore National Laboratory, Livermore, CA 94550, USA

Abstract

A set of implicit methods are proposed for a third-order hierarchical WENO reconstructed discontinuous Galerkin method for compressible flows on 3D hybrid grids. An attractive feature in these methods are the application of the Jacobian matrix based on the P1 element approximation, resulting in a huge reduction of memory requirement compared with DG (P2). Also, three approaches — analytical derivation, divided differencing, and automatic differentiation (AD) are presented to construct the Jacobian matrix respectively, where the AD approach shows the best robustness. A variety of compressible flow problems are computed to demonstrate the fast convergence property of the implemented flow solver. Furthermore, an SPMD (single program, multiple data) programming paradigm based on MPI is proposed to achieve parallelism. The numerical results on complex geometries indicate that this low-storage implicit method can provide a viable and attractive DG solution for complicated flows of practical importance.

1. Introduction

The discontinuous Galerkin (DG) methods [1–26] have recently become popular for the solution of systems of conservation laws. The DG methods were first introduced for the solution of neutron transport equations [27], and nowadays they are widely used in computational fluid dynamics (CFD), computational acoustics, and computational magneto-hydrodynamics (MHD). A number of attractive features of the DG methods are listed in Ref. [28]. However, the DG methods also have their own weaknesses. Indeed, compared to the finite element (FE) and finite volume (FV) methods, the DG methods require solutions of systems of equations with more unknowns for the same grids. Consequently, DG are recognized as expensive in terms of both computational costs and storage requirements.

In recent years, a number of so-called reconstruction DG schemes are proposed in order to reduce the high costs associated with the standard DG methods, like the recovery-based RDG (Pn,Pm) schemes (n ≤ m) by Dumbser et al. [29–31], the hybrid HWENO + DG schemes by Balsara et al. [32], the least-squares reconstruction DG schemes by Luo et al. [33–37,28], and the class of Green-Gauss reconstruction hybrid DG/FV schemes by Zhang et al. [38,39]. All of these schemes are able to improve the spatial accuracy of the underlying DG methods without significant extra cost in storage and computing time. On the other side, in order to achieve fast convergence for DG methods, implicit time integration is required. Unfortunately, many high-order implicit DG methods [40,41,5,42,9,7,43,44] require a considerable amount of memory to store the Jacobian matrix. Indeed, it is our belief that a lack of efficient solvers is one of the reasons that the application of the DG methods for engineering-type problems does not exist.

Recently the present authors proposed an implicit WENO reconstruction-based DG method for compressible flows on tetrahedral grids [45,46]. In this method, computation of the Jacobian matrix is based on the underlying DG (P1) method only. The most attractive feature of this implicit method is its low-storage requirement due to its DG (P1)-like linear system of equations, which requires less than 1/6 of the memory of the implicit DG (P2) method to achieve the same order of accuracy. Furthermore, in order to alleviate the excessive human labor for constructing a well approximated Jacobian matrix, the present authors introduced an implicit method based on automatic differentiation for the WENO reconstruction-based DG method on tetrahedral grids [47]. The resulting implicit method is highly robust and efficient according to a variety of test cases. In the present work, this method is extended to 3D hybrid grids. Besides, the performance of three approaches for constructing the Jacobian matrix: (1) analytical
derivation, (2) divided differencing, and (3) automatic differentiation is analyzed, respectively. A parallel strategy is proposed and implemented based on a message passing interface (MPI) programming paradigm. A variety of compressible flow problems for a wide range of flow conditions in 3D configurations are computed to demonstrate the performance of the developed implicit methods.

The remainder of this paper is organized as follows. The governing equations are described in Section 2. The discontinuous Galerkin discretization and the hierarchical WENO reconstruction schemes are briefly introduced in Sections 3 and 3.2. The implicit time integration method is given in Section 4. A set of methods to construct the Jacobian matrix are presented in Section 5. The parallelization strategy is illustrated in Section 6. The numerical experiments are reported in Section 7. Finally, concluding remarks are given in Section 8.

2. Governing equations

The Navier–Stokes equations governing unsteady compressible viscous flows can be expressed as

\[
\frac{\partial \mathbf{U}(x,t)}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U}(x,t))}{\partial x} = \frac{\partial \mathbf{G}(\mathbf{U}(x,t), \nabla \mathbf{U}(x,t))}{\partial x}
\]  

(1)

where the summation convention has been used. The conservative variable vector \( \mathbf{U} \), advective flux vector \( \mathbf{F} \), and viscous flux vector \( \mathbf{G} \) are defined by

\[
\mathbf{U} = \begin{pmatrix} \rho \\ \rho u_i \\ \rho e \end{pmatrix}, \quad \mathbf{F}_j = \begin{pmatrix} \rho u_j \\ \rho u_i u_j + p \delta_{ij} \\ u_j (\rho e + p) \end{pmatrix}, \quad \mathbf{G}_j = \begin{pmatrix} 0 \\ \tau_{ij} \\ q_j \end{pmatrix}
\]  

(2)

Here \( \rho, p, \) and \( e \) denote the density, pressure, and specific total energy of the fluid, respectively, and \( u_i \) is the velocity of the flow in the coordinate direction \( x_i \). The pressure can be computed from the equation of state

\[
p = (\gamma - 1) \rho \left( e - \frac{1}{2} u^2 \right)
\]  

(3)

which is valid for perfect gas. The ratio of the specific heats \( \gamma \) is assumed to be constant and equal to 1.4. The viscous stress tensor \( \tau_{ij} \) and heat flux vector \( q_j \) are given by

\[
\tau_{ij} = \mu \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \frac{2}{3} \mu \frac{\partial u_k}{\partial x_k} \delta_{ij}, \quad q_j = \frac{1}{\gamma - 1} \frac{\mu}{\rho} \frac{\partial T}{\partial x_j}
\]  

(4)

In the above equations, \( T \) is the temperature of the fluid, \( Pr \) the laminar Prandtl number, which is taken as 0.7 for air. \( \mu \) represents the molecular viscosity, which can be determined through Sutherland’s law

\[
\mu = \frac{\mu_0}{\rho_0} \left( \frac{T}{T_0} \right)^{\frac{1}{2}} \left( T_0 + S \right) \left( \frac{T}{T_0} + S \right)
\]  

(5)

\( \mu_0 \) denotes the viscosity at the reference temperature \( T_0 \) and \( S = 110 K \). The temperature of the fluid \( T \) is determined by

\[
T = \frac{\gamma p}{\rho}
\]  

(6)

Neglecting viscous effects, the left-hand-side of Eq. (1) represents the Euler equations governing unsteady compressible inviscid flows.

3. Reconstructed discontinuous Galerkin method

3.1. Discontinuous Galerkin spatial discretization

Eq. (1) can be discretized using a discontinuous Galerkin finite element formulation. First, we assume that the domain \( \Omega \) is subdivided into a collection of non-overlapping arbitrary elements \( \Omega_e \) in 3D, and introduce the following Sobolev space \( V_h^p \)

\[
V_h^p = \left\{ v_h \in [L^2(\Omega)]^m : v_h|_{\Omega_e} \in [V^p] \quad \forall \Omega_e \in \Omega \right\}
\]  

(7)

which consists of discontinuous vector polynomial functions of degree \( p \), and where \( m \) is the dimension of the unknown vector and \( V_p \) is the space of all polynomials of degree \( \leq p \). To formulate the discontinuous Galerkin method, we introduce the following weak formulation, which is obtained by multiplying Eq. (1) by a test function \( W \), integrating over an element \( \Omega_e \), and then performing an integration by parts: find \( \mathbf{U} \in V_h^p \) such as

\[
\frac{d}{dt} \int_{\Omega_e} \mathbf{U} \cdot \mathbf{w} \, d\Omega + \int_{\Gamma_e} \mathbf{F}_i(\mathbf{U}) \cdot \mathbf{n} \cdot \mathbf{w} \, d\Gamma - \int_{\Gamma_e} \mathbf{G}_i(\mathbf{U}) \frac{\partial \mathbf{U}}{\partial n} \, d\Gamma = \int_{\Omega_e} \mathbf{G}_i(\mathbf{U}, \nabla \mathbf{U}) \cdot \frac{\partial \mathbf{w}}{\partial n} \, d\Omega + \int_{\Gamma_e} \mathbf{F}_i(\mathbf{U}, \nabla \mathbf{U}) \cdot \frac{\partial \mathbf{w}}{\partial n} \, d\Gamma
\]  

(8)

where \( \mathbf{U} \) and \( W \) are represented by piecewise polynomial functions of degree \( p \), which are discontinuous between the cell interfaces, and \( \mathbf{n} \) the unit outward normal vector to the \( \Gamma_e \) : the boundary of \( \Omega_e \). Assume that \( B_i \) is the basis of polynomial function of degrees \( p \), this is then equivalent to the following system of \( N \) equations,

\[
\frac{d}{dt} \int_{\Omega_e} \mathbf{U} \mathbf{B}_i \, d\Omega + \int_{\Gamma_e} \mathbf{F}_i(\mathbf{U}) \cdot \mathbf{n} \mathbf{B}_i \, d\Gamma - \int_{\Gamma_e} \mathbf{G}_i(\mathbf{U}) \frac{\partial \mathbf{B}_i}{\partial n} \, d\Gamma = \int_{\Omega_e} \mathbf{G}_i(\mathbf{U}, \nabla \mathbf{U}) \cdot \frac{\partial \mathbf{B}_i}{\partial n} \, d\Omega + \int_{\Gamma_e} \mathbf{F}_i(\mathbf{U}, \nabla \mathbf{U}) \cdot \frac{\partial \mathbf{B}_i}{\partial n} \, d\Gamma \quad 1 \leq i \leq N
\]  

(9)

where \( N \) is the dimension of the polynomial space. Since the numerical solution \( \mathbf{U} \) is discontinuous between element interfaces, the interface fluxes are not uniquely defined. The flux function \( \mathbf{F}_i(\mathbf{U}) \cdot \mathbf{n} \) appearing in the second terms of Eq. (9) is replaced by a numerical Riemann flux function \( \mathbf{H}_i(\mathbf{U}_h^0, \mathbf{U}_h^0, \mathbf{U}_h^0) \) where \( \mathbf{U}_h^0 \) is the conservative state vectors at the left and right side of the element boundary. This scheme is called discontinuous Galerkin method of degree \( p \), or in short notation DG \( (p) \) method. By simply increasing the degree \( p \) of the polynomials, the DG methods of corresponding higher order are obtained. In the present work, the inviscid flux is evaluated by the HLLC [48] scheme and the viscous flux by the Bassi–Rebay II scheme [5], respectively. For nodal discontinuous Galerkin methods, numerical polynomial solutions \( \mathbf{U}_h \) in each element are expressed using either standard Lagrange finite element or hierarchical node-based basis like

\[
\mathbf{U}_h = \sum_{i=1}^{N} \mathbf{U}_i \mathbf{B}_i(\mathbf{x})
\]  

(10)

where \( \mathbf{B}_i \) is the finite element basis function. The resulting unknowns to be solved are the variables at the nodes \( \mathbf{U}_i \). In the DG method of our work, the numerical polynomial solutions are represented using a Taylor series expansion at the cell centroid and normalized in order to improve the conditioning of the system matrix Eq. (9). For example, the linear polynomial \( P_1 \) solutions of the underlying DG \( (P_1) \) method used in the present work, consist of cell-averaged values \( \bar{\mathbf{U}} \) and their normalized first derivatives \( \mathbf{U} = \frac{\partial \mathbf{U}}{\partial x} \Delta x, \mathbf{U}_i = \frac{\partial \mathbf{U}}{\partial y} \Delta y, \mathbf{U}_i = \frac{\partial \mathbf{U}}{\partial z} \Delta z \) at the center of the cell, as expressed below,

\[
\mathbf{U}_h^{(i)} = \mathbf{U} \mathbf{B}_i + \mathbf{U}_j \mathbf{B}_j + \mathbf{U}_k \mathbf{B}_k + \mathbf{U}_l \mathbf{B}_l
\]  

(11)

where the four basis functions are as below.
\[B_1 = 1 \quad B_2 = \frac{x - x_j}{\Delta x} \quad B_3 = \frac{y - y_j}{\Delta y} \quad B_4 = \frac{z - z_j}{\Delta z}\]  

(12)

where \(\Delta x = 0.5(x_{max} - x_{min})\), \(\Delta y = 0.5(y_{max} - y_{min})\), \(\Delta z = 0.5(z_{max} - z_{min})\). \(x_{max}, y_{max}, z_{max}\) and \(x_{min}, y_{min}, z_{min}\) are the maximum and minimum vertex coordinates of the cell \(\Omega_j\), respectively. The above normalization is especially important to make the system matrix less stiff for higher-order discontinuous Galerkin approximations.

This formulation has a number of attractive, distinct, and useful features. First, cell-averaged variables and their derivatives are handily available in this formulation. This makes the implementation of both in-cell and inter-cell reconstruction schemes straightforward and simple [33,35,49,39,50]. Secondly, the Taylor basis is hierarchic, which greatly facilitates the implementation of p-multigrid methods [51,52] and p-refinement. Thirdly, the same basis functions are used for any shapes of elements: tetrahedron, pyramid, prism, and hexahedron. This makes the implementation of DG methods on arbitrary grids straightforward.

### 3.2. Hierarchical WENO reconstruction scheme

A third-order hierarchical WENO reconstruction scheme based on the reconstructed discontinuous Galerkin method [34,50] designed by the authors earlier for tetrahedral grids [28] is extended to 3D hybrid grids in this work. This scheme adopts a hierarchical reconstruction strategy [53] (successively from high order to low order), where the second and first derivatives are reconstructed in a hierarchical manner.

Firstly, the six normalized second derivatives \(U^2_{ij} = \frac{\partial^2 u}{\partial x_i \partial x_j}\) are computed using a least-squares method [36] from the underlying linear polynomial \((P_1)\) discontinuous Galerkin solution, and the reconstructed quadratic polynomial \((P_2)\) solution is expressed as follows:

\[U^2_{ij} = U^1_{ij} + U^2_{ij}B_5 + U^2_{ij}B_6 + U^2_{ij}B_7 + U^2_{ij}B_8 + U^2_{ij}B_9 + U^2_{ij}B_{10}\]  

(13)

where the six additional basis functions are

\[B_5 = \frac{B^2_2}{2} \int_{\Omega_i} B^2_2 d\Omega \quad B_6 = \frac{B^2_1}{2} \int_{\Omega_i} B^2_1 d\Omega \quad B_7 = \frac{B^2_2}{2} \int_{\Omega_i} B^2_2 d\Omega \quad B_8 = B_5B_3 - \frac{1}{2} \int_{\Omega_i} B_5B_3 d\Omega \quad B_9 = B_5B_4 - \frac{1}{2} \int_{\Omega_i} B_5B_4 d\Omega \quad B_{10} = B_5B_4 - \frac{1}{2} \int_{\Omega_i} B_5B_4 d\Omega\]  

(14)

The final second derivatives are then obtained using a WENO reconstruction as expressed below, which is necessary to ensure the linear stability of the RDG method on unstructured elements,

\[\frac{\partial^2 U}{\partial x_i \partial x_j}^{\text{WENO}} = \sum_{k=1}^{N_n} W_k \frac{\partial^2 U}{\partial x_i \partial x_j}^{R}\]  

(15)

where \(N_n\) denotes the number of face-neighboring cells (equals to 4, 5, 5, and 6 for a tetrahedron, pyramid, prism and hexahedron, respectively). The procedure of computing the normalized nonlinear weights \(w_{ij}^{(2)}\) is described in Ref. [36]. The resulting RDG \((P_1P_2)\) method is referred to as WENO \((P_1P_2)\).

Secondly, the first derivatives are reconstructed using a WENO reconstruction in order to eliminate non-physical oscillations in the vicinity of strong discontinuities and thus maintain the nonlinear stability,

\[\frac{\partial U}{\partial x_i}^{\text{WENO}} = \sum_{k=1}^{N_n} W_k \frac{\partial U}{\partial x_i}^{R}\]  

(16)

where the procedure of computing the normalized nonlinear weights \(w_{ij}^{(1)}\) are described in Ref. [28]. The resulting RDG \((P_1P_2)\) method based on this hierarchical WENO reconstruction is termed as HWENO \((P_1P_2)\). The present choice of reconstruction stencils is symmetric, and compact, as only von Neumann neighbors are involved, as demonstrated in Fig. 1. This means that no additional data structure is required for our HWENO \((P_1P_2)\) scheme.

In the HWENO \((P_1P_2)\) RDG method, the reconstructed quadratic polynomial solution is then used to compute the domain and boundary integrals of the underlying DG \((P_1)\) method in Eq. (9). As demonstrated in Ref. [28], this resulting HWENO \((P_1P_2)\) RDG method is also able to achieve the designed third-order of accuracy for smooth flows at a moderate increase of computing costs in comparison with the underlying DG \((P_1)\) method. The extra costs are mainly due to the reconstruction, which is relatively cheap in comparison to the evaluation of fluxes, and an extra Gauss quadrature point, which is required to calculate the domain integrals for the tetrahedral cell (5 quadrature points), as demonstrated in Table 1. In comparison to DG \((P_2)\), this represents a significant saving in terms of flux evaluations. Most importantly, the number of degrees of freedom is significantly reduced, which leads to a significant reduction in memory requirements, and from which implicit methods will benefit tremendously. As a result, the implicit methods for HWENO \((P_1P_2)\) requires a much lower storage than DG \((P_2)\): 400 versus 2500 words per elemental implicit diagonal matrix (IDM), while both can achieve the same order of accuracy!

### 4. Implicit temporal discretization

The spatial discretization of the governing equations leads to a system of ordinary differential equations (ODEs) in time and Eq. (9) can be written in an elemental semi-discrete form as

\[M \frac{dU}{dt} = R(U)\]  

(17)

where \(U = [u_1, u_2, \ldots, u_n]^{T}\) is the global solution vector of \(N_{\text{deg}} \times \text{Netot} \times \text{Nelem} \) DOFs to be evolved in time. By applying the backward Euler scheme to Eq. (17), one obtains

\[M \frac{(U^{n+1} - U^n)}{\Delta t} = R(U^{n+1})\]  

(18)

which is a system of nonlinear equations for \(U^{n+1}\). In order to solve this type of equations, we can linearize \(R\) with respect to \(U\) at the current time-step

\[R(U^{n+1}) \approx R(U^n) + \frac{\partial R}{\partial U}(U^{n+1} - U^n)\]  

(19)

where \(\left(\frac{\partial R}{\partial U}\right)^n\) is the Jacobian matrix of the system, and denoted symbolically as \(J(U^n)\). If we replace the right-hand-side term in Eq. (18) with Eq. (19), we can get a delta form of the linear system of equations as follows:

\[A\Delta U^n = \left(\frac{M}{\Delta t} - \frac{\partial R}{\partial U}\right)^n \Delta U^n = R(U^n)\]  

(20)

where \(A\) is the left-hand-side matrix, and \(\Delta t\) is the time increment and \(A\Delta U^n = U^{n+1} - U^n\) is the solution difference between time level \(n\) and \(n + 1\). In the present flow solver, two algorithms are used for the solution of the resulting linear system of equations. One is the SGS \((k)\) (Symmetric Gauss-Seidel) method [54], where \(k\) is a pre-set number of sub-iterations. The other one is the LU-SGS (lower-upper SGS) preconditioned GMRES (generalized minimal residual) method [45–47,55], termed as GMRES + LU-SGS. These two methods are well accepted and the implementation details can be found in referred articles above. Also note that the GMRES [56] method is among the most popular and efficient iterative
algorithms and employed by various authors [57,40,41,58,42,13,60,61,8,44].

Table 1
| Cost for DG (P1), HWENO (P1P2) and DG (P2) on a tetrahedral cell. |
|------------------------|------------------------|------------------------|
| NQ for \( \int \mathbf{f} d\Omega \) | DG (P1) | HWENO (P1P2) | DG (P2) |
| NQ for \( \int \mathbf{g} d\Omega \) | No | Yes | No |
| Reconstruction | \( O(\delta^2) \) | \( O(\delta^3) \) | \( O(\delta^4) \) |
| Unit mem. for IDM | 400 | 400 | 2500 |

5. Construction of the Jacobian matrix

In general, four methods are available to construct the Jacobian matrix of the linear system of equations: (1) based on analytical derivation; (2) based on numerical differentiation; (3) based on automatic differentiation; and (4) based on symbolic mathematical software (this is not implemented due to the time-consuming manual operation). The first three methods are to be discussed in detail, respectively.

5.1. Based on analytical derivation

The method based on analytical derivation requires the manual implementation of analytic derivative formulae, which can typically result in very efficient code. However, this method is always tedious, error-prone, and requires great manpower, if no simplification of the system is made. In the case of the HWENO (P1P2) RDG method, the exact Jacobian matrix is very hard to obtain, since the HWENO (P1P2) reconstruction is highly nonlinear in nature, and thus the linearization of such process might not be easily conducted in an explicit layout. Even in the standard DG method, exact linearization of the viscous flux scheme, e.g., the BR2 scheme, is not trivial work. Thus more than usual, approximate Jacobian matrix is used instead of the exact one in implicit DG methods [48,41,58,42,52,62,13,63,44]. In the present work, the Jacobian matrix for the HWENO (P1P2) RDG method can be derived based on the approximate linearization of the underlying DG (P1) method, that is, \( J_{\text{HWENO}(P1P2)} \approx J_{\text{DG}(P1)} \). More specifically, the following two approximations are adopted in face integrals and domain integrals, respectively,

\[
\frac{\partial \mathbf{H}_i \left( \mathbf{U}_i^k, \mathbf{U}_j^k, \mathbf{n}_j \right)}{\partial \mathbf{U}_i} = \frac{\partial \mathbf{H}_i \left( \mathbf{U}_i^k, \mathbf{U}_j^k, \mathbf{n}_j \right)}{\partial \mathbf{U}_j^k} \frac{\partial \mathbf{U}_j^k}{\partial \mathbf{U}_i} \approx \frac{\partial \mathbf{H}_i \left( \mathbf{U}_i, \mathbf{U}_j, \mathbf{n}_j \right)}{\partial \mathbf{U}_i} \tag{21}
\]

\[
\frac{\partial \mathbf{F}_i \left( \mathbf{U}_i^k \right)}{\partial \mathbf{U}_i} = \frac{\partial \mathbf{F}_i \left( \mathbf{U}_i^k \right)}{\partial \mathbf{U}_i^k} \frac{\partial \mathbf{U}_i^k}{\partial \mathbf{U}_i} \approx \frac{\partial \mathbf{F}_i \left( \mathbf{U}_i \right)}{\partial \mathbf{U}_i} \tag{22}
\]

where \( \mathbf{U}_i^k \) denotes the HWENO (P1P2) reconstructed solution. Furthermore, for the inviscid part, the HLLC flux scheme is linearized approximately by assuming the wavespeed as constant in face integrals, which is called the frozen wavespeed version of implicit HLLC scheme in Ref. [48]. For the viscous part, the BR2 scheme is linearized by treating the local lift operator in face integrals, the global lift operator in domain integrals, and the molecular viscosity as constant, respectively.

5.2. Based on numerical differentiation

The method based on numerical differentiation, more specifically, backward divided differencing (DD), operates on some truncation of the Taylor series. This method is easy to implement by evaluating the residual vector using perturbations of the solution vector, as expressed below:

\[
J_{ij} = \frac{\partial R_i}{\partial U_j} \approx \frac{R_i(\mathbf{U} + \mathbf{e} \cdot \mathbf{e}_j) - R_i(\mathbf{U})}{\mathbf{e}}, \quad \mathbf{e} \in \mathbb{R} \quad \text{a small number} \tag{23}
\]

where subscript \( ij \) = 1, 2, ..., \( (N\text{degr} \times \text{Netot}) \) (20 for HWENO (P1P2)). In addition, the computational mesh needs to be divided into colored groups, which is required to confine elemental perturbation within the element itself. In another word, two face-neighboring cells cannot be in the same group. This DD-based method is very easy to implement, and the result can approach closely to the exact Jacobian matrix with a properly estimated perturbation variable \( \mathbf{e} \) (usually \( 10^{-7} - 10^{-8} \) for smooth flows).

However, there are two issues with this method. For the first, \( \mathbf{e} \) is in fact problem-dependent, and the method will become unstable where flow conditions change drastically, e.g., the shock waves, thus can result in a sudden breakdown of the linear solver. For the second, this method is very computationally intensive: it requires \( k \times N\text{degr} \times \text{Netot} \) times in each Newton iteration to evaluate the residual vector, (that is 20k for DG (P1) and HWENO (P1P2), and 50k for DG (P2)), where \( k \) is the number of colored groups (\( k = 6-8, \) depending on meshes), and thus resulting in a highly expensive method.
5.3. Based on automatic differentiation

The method based on automatic differentiation (AD) is to compute the Jacobian matrix using the AD-generated source code. Automatic differentiation is a technology for automatically augmenting computer programs, including arbitrarily complex simulations, with statements for the computation of derivatives, also known as sensitivities.

By using an AD tool, e.g., TAPENADE [64] as adopted in the present work, the labor for manual coding can be significantly reduced, which otherwise can be very complicated, tedious and error-prone in the discontinuous Galerkin context if operated manually, depending on the complexity of the numerical functions. Similar to divided differencing, automatic differentiation requires only the source program \( C \). But instead of executing \( C \) on different sets of inputs, the AD tool builds a new, augmented code \( C' \) that computes the analytical derivatives along with the source program. This new program is called the differentiated program. Each time the source program holds some value \( v \), the differentiated program holds an additional value \( dv \), which is the differential of \( v \).

In the case of evaluation of the R.H.S residual vector in the present work, the reconstruction, which is before computing the face and domain integrals by using the reconstructed polynomial solution, is not included in the AD source program, due to the structure of legacy code. Thus like the method based on analytical derivation as introduced above, the source program of the underlying DG (P1) method is provided for TAPENADE. For the inviscid part, the AD process results in an differentiated program in which the HLLC flux functions are exactly linearized. However, numerical tests conducted by Batten et al. [48] showed that the exact implicit HLLC scheme provided no further improvement than the frozen wave-speed version of this scheme for convergence in terms of computing time, which is again confirmed in the present work. For the viscous part, the AD process of face integrals results in an exact linearization of the diffusion terms, including the local lift operator of the BR2 scheme. But in the AD process of domain integrals, the global lift operator of the BR2 scheme is treated as constant. This simplification is made in order to offset the overhead of doubly computing the face-based local lift operator to formulate the cell-based global lift operator. With the attractive feature described above, the AD-based method can also largely reduce the workload for code maintenance in case the underlying numerical flux schemes are updated, as the programmer only needs to conduct the AD process with the updated source program \( C \) and use the generated code \( C' \) back in the solver code.

6. Parallelization

The compactness of the HWENO (P1,P2) RDG method makes it ideally suited for parallel computing. In the present work, an SPMD (single program, multiple data) programming paradigm based on the MPI library is adopted to achieve parallelism, the METIS library [65] is used for the partitioning of a grid into sub-domain grids of approximately the same size. Two examples are illustrated in Fig. 2. The first one is a prismatic + hexahedral hybrid grid for flow around a circular cylinder, which is split into 16 sub-domains as shown in Fig. 2a. The second one is a tetrahedral grid for flow over a wing/pylon/finned-store configuration, which is split into 64 sub-domains as shown in Fig. 2b. The communication in parallel mode is managed by the necessary standard MPI commands like non-blocking send, nonblocking receive and wait commands. In the present work, parallelization is implemented for both the explicit and implicit methods.

7. Numerical examples

Computations on a series of well-documented test cases for the compressible inviscid and viscous flows are carried out in this section. In these test cases, the serial computations are conducted on a Dell Precision T7400 personal computer (2.98 GHz Xeon CPU with 18 GBytes memory), and the parallel computations are performed on a cluster (AMD Opteron 6128 8-core 2.0 GHz processor with 32 GBytes memory for each compute-node).

The following \( L^2 \) norm of the entropy production is used as the error measurement for the steady-state inviscid flow problems

\[
\|\varepsilon\|_{L^2} = \sqrt{\int_{\Omega} \varepsilon^2 \, d\Omega} = \sqrt{\sum_{i=1}^{Nelem} \int_{\Omega_i} \varepsilon^2 \, d\Omega}
\]

where the entropy production \( \varepsilon \) is defined as

\[
\varepsilon = \frac{S - S_\infty}{S_\infty} = \frac{p}{p_\infty} \left( \frac{\rho_\infty}{\rho} \right)^\gamma - 1
\]

Note that the entropy production, where the entropy is defined as \( S = (p/\rho)^\gamma \), is a very good criterion to measure the accuracy of numerical solutions, since the flow under consideration is isentropic.

In the present test cases, the convergence speedup factor for the implicit methods is assessed according to the baseline results obtained by the three-stage TVD Runge–Kutta (TVDRK3) explicit time stepping scheme. For 2D illustration of a cluster of variables...
on the surface of solid body, e.g., extracted surface pressure coefficients on a cut plane, the variables are computed at the two nodes of each cell face that intersect with the cut plane, and plotted by a straight line. This is the most accurate way to represent the $P_1$ solution, as the solution is linear on each cell face and multiple values can exist across the cell interface due to the discontinuous representation of DG solution. Some phrase abbreviations for describing a computational grid are defined as follows: Nelem, number of total elements of a grid; Ntet, number of tetrahedral elements; Npyra, number of pyramidal elements; Npris, number of prismatic elements; Nhexa, number of hexahedral elements; Nfac, number of total boundary faces; Npoin, number of grid vertex node.

### 7.1. Subsonic flow past a circular cylinder

This is a well-known test case: inviscid subsonic flow past a circular cylinder at a Mach number of $M_1 = 0.38$. This test case is chosen to test the accuracy of the WENO ($P_1$,$P_2$) methods and the performance of the implicit methods on hybrid grids. This is a 3D simulation of the 2D problem. Computation is conducted on a series of four successively refined prismatic + hexahedral hybrid grids, having $(16 + 8) \times 4$ (Level-1), $(32 + 16) \times 8$ (Level-2), $(64 + 32) \times 16$ (Level-3) and $(128 + 64) \times 32$ (Level-4) cells in the $x$-$y$ plane and 1 cell in the $z$-direction, as shown in Fig. 3a–d. Numerical solutions to this problem are computed using DG ($P_1$) and WENO ($P_1$,$P_2$) on these grids to obtain a quantitative measurement of the

![Fig. 3.](image-url)
discretization errors, which are presented in Tables 2 and 3. They show the grid size, the \( L^2 \) error of the solutions, and the piece-wise order of convergence. Fig. 3e–h, and Fig. 3i–l show the computed Mach number contours in the flow field obtained by the DG (P1) and WENO (P1P2) solutions on the four grids, respectively. One can see that the WENO (P1P2) solutions are more accurate than the DG (P1) solutions on the first three grids. Both the DG (P1) and WENO (P1P2) solutions are virtually identical on the finest grid.

![Fig. 3 (continued)](image)

**Table 2**
The circular cylinder case on prismatic/hexahedral hybrid grids: DG (P1).

<table>
<thead>
<tr>
<th>No. of cells</th>
<th>Cell size</th>
<th>( L^2 ) error</th>
<th>Order (slope)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(16 + 8) × 4</td>
<td>h</td>
<td>-0.12907E+01</td>
<td></td>
</tr>
<tr>
<td>(32 + 16) × 8</td>
<td>h/2</td>
<td>-0.19912E+01</td>
<td>2.326</td>
</tr>
<tr>
<td>(64 + 32) × 16</td>
<td>h/4</td>
<td>-0.27710E+01</td>
<td>2.593</td>
</tr>
<tr>
<td>(128 + 64) × 32</td>
<td>h/8</td>
<td>-0.35798E+01</td>
<td>2.697</td>
</tr>
</tbody>
</table>

**Table 3**
The circular cylinder case on prismatic/hexahedral hybrid grids: WENO (P1P2).

<table>
<thead>
<tr>
<th>No. of cells</th>
<th>Cell size</th>
<th>( L^2 ) error</th>
<th>Order (slope)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(16 + 8) × 4</td>
<td>h</td>
<td>-0.12000E+01</td>
<td></td>
</tr>
<tr>
<td>(32 + 16) × 8</td>
<td>h/2</td>
<td>-0.23306E+01</td>
<td>3.762</td>
</tr>
<tr>
<td>(64 + 32) × 16</td>
<td>h/4</td>
<td>-0.36842E+01</td>
<td>4.500</td>
</tr>
<tr>
<td>(128 + 64) × 32</td>
<td>h/8</td>
<td>-0.46498E+01</td>
<td>3.219</td>
</tr>
</tbody>
</table>
However, WENO (P₁P₂) does yield a slightly more accurate solution than DG (P₁) at the same grid resolution if we compare the $L^2$ errors in Tables 2 and 3. Seen from Fig. 4a, DG (P₁) and WENO (P₁P₂) each achieves an averaged slope of over 2 and 3, respectively. Fig. 4b illustrates that WENO (P₁P₂) requires significantly fewer degrees of freedom than DG (P₁) for the same accuracy. Fig. 5a and b shows a comparison of convergence histories versus time steps and CPU time on the Level-3 grid, respectively for implicit DG (P₁) and WENO (P₁P₂) using the Jacobian matrix based on analytical derivation. The implicit method converges in nearly the same number of time steps and CPU time for both DG (P₁) and WENO (P₁P₂) solutions, demonstrating the order independence of this implicit method. The performance of the implicit DG (P₁) and WENO (P₁P₂) methods is also carried out on the Level-4 grid, as displayed in Fig. 6a and b. Again, the similar order independence of the implicit method is observed.

7.2. Subsonic flow past a sphere

In this test case, an inviscid subsonic flow past a sphere at a free-stream Mach number of $M_1 = 0.5$ is chosen to assess the accuracy of WENO (P₁P₂) and the performance of the implicit methods on 3D configurations. A sequence of three successively refined tetrahedral grids for computation are shown in Fig. 7a–c, respectively: Level-1 (Nelem = 535, Npoin = 167, Nafac = 244), Level-2 (Nelem = 2426, Npoin = 589, Nafac = 640), and Level-3 (Nelem = 16,467, Npoin = 3425, Nafac = 2372). The cell size is halved between two consecutive grids. Note that only a quarter of the grids are shown for clarity.
of the configuration is modeled due to symmetry of the problem. Fig. 7d–f and g–i display the computed Mach number contours in the flow field by DG (P₁) and WENO (P₁P₂) solutions, respectively. Fig. 8a and b show the $L^2$ errors of numerical solutions versus cell size and No. DOFs respectively. The results obtained by DG (P₁) and WENO (P₁P₂), perhaps not as impressive as those shown in the previous test case likely due to the tetrahedral grid quality, do indicate that DG (P₁) and WENO (P₁P₂) exhibit a $O(h^3)$ and $O(h^3)$ order of convergence on smooth solutions, respectively. Fig. 9a and b displays a comparison of convergence histories versus time steps and CPU time between explicit and implicit methods, using WENO (P₁P₂) on the Level-1, Level-2 and Level-3 grids respectively. The implicit method uses the Jacobian matrix based on analytical derivation, which obtains the convergence in nearly the same number of time steps on all the three grids as shown in Fig. 9a, demonstrating the order independence of this implicit method. The implicit method is over 30 times faster than its explicit TVDRK3 counterpart as shown in Fig. 9b.

7.3. Transonic flow over the ONERA M6 wing

An inviscid transonic flow over the ONERA M6 wing at $M_\infty = 0.84$ and $\alpha = 3.06$ is considered in this case in order to assess the performance of HWENO (P₁P₂) and the implicit methods at the presence of strong discontinuities. A tetrahedral grid (Nelem = 95,266, Npoin = 18,806, Nfac = 10,579) is used in

---

Fig. 7. (a–c) A sequence of three successively refined tetrahedral grids. Computed Mach number contours obtained by (d–f) DG (P₁) and (g–i) WENO (P₁P₂) solutions, for inviscid subsonic flow past a sphere at $M_\infty = 0.5$. 
computation, as the surface triangular mesh is shown in Fig. 10a. Fig. 10b shows the computed surface pressure contours on the upper wing. The computed pressure coefficients are compared with experimental data [66] at six span-wise locations in Fig. 11a–f. The results conform closely with the experimental data, except at the root stations as shown in Fig. 11a and b, due to lack of viscous effects. The leading edge suction peak is extremely well captured in spite of the low grid resolution. The shocks are virtually captured within one cell without oscillations, clearly demonstrating the high accuracy and non-oscillatory property of our HWENO (P1\(^2\)) reconstruction scheme. Fig. 12a and b displays the convergence histories versus time steps and CPU time respectively, obtained by the implicit method, in which the Jacobian matrix is based on analytical derivation. This is a one-minute job by taking advantage of parallel computing on 128 CPU processors! Besides, note that the residual level would not further drop down after a decrease of 3 or 4 orders of magnitude in this test case, although the flow field has reached the steady state. This phenomenon is sometimes called “convergence stall”, which could be observed for problems with strong discontinuities or sharp edges, especially when nonlinear approaches like WENO schemes are used.

7.4. Transonic flow over the wing/pylon/finned-store configuration

An inviscid transonic flow over the wing/pylon/finned-store configuration at a free-stream Mach number of \( M_\infty = 0.95 \) and attack angle of \( \alpha = 0^\circ \) is considered in this test case, in order to...
assess the performance of HWENO (P1P2) and the implicit methods for complex geometric configurations. The detailed description of this configuration is reported by Ref. [67]. The configuration consists of a clipped delta wing with a 45° sweep comprised from a constant NACA 64010 symmetric airfoil section. The wing has a root chord of 15 inches, a semi-span of 13 inches, and a taper ratio of 0.134. The pylon is located at the midspan station and has a cross-section characterized by a flat plate closed at the leading and trailing edges by a symmetrical olive shape. The width of the pylon is 0.294 inches. The four fins on the store are defined by a constant NACA 0008 airfoil section with a leading-edge sweep of 45° and a truncated tip. Computation is conducted on a tetrahedral grid (Nelem = 319,134, Npoin = 147,289, Nafac = 28,738), as the surface meshes are shown in Fig. 13a and b. Fig. 13c and d show the computed pressure contours on the upper and lower wing surfaces, respectively. The computed pressure coefficient distributions exhibit a consistent convergence in the flow field. Fig. 17a and b display the convergence histories of versus time steps and CPU time respectively for both the explicit and implicit methods. The GMRES method is chosen as the linear solver in the implicit method for viscous flows. The computational domain is bounded from −0.5 to 1.0 along the x-direction, from 0 to 1.0 along the y-direction, and from 0 to 0.1 along the z-direction, and the no-slip surface starts at point (0.0, 0.2, 0.0) and extends to (0.0, 0.2, 0.1). A slip condition is prescribed along the bottom side of the domain for x ∈ [−0.0, 0.0] with ν = 0. Symmetry conditions are prescribed for the front and back boundary with w = 0. The characteristic boundary is prescribed to the left side (x = −0.5), top side (y = 1), and right side (x = 1).

Firstly, computation is conducted on a tetrahedral grid (Ntet = 47,536) as shown in Fig. 16a. Fig. 16b renders a closer view at the near-wall region, which depicts the sparsity of the grid resolution in the boundary layer. Fig. 16c compares the computed distributions obtained by DG (P1) and WENO (P1P2) solutions with the analytical solution. They are highly accurate on this grid. The difference is only discernible if one takes a zoom-in observation, e.g., for x ∈ [0.5, 0.6] in Fig. 16d, where WENO (P1P2) presents a slightly better prediction than DG (P1). Fig. 16e and f compare the computed velocity profiles with the analytical solutions at the three downstream locations, where both the DG (P1) and WENO (P1P2) solutions exhibit a consistent convergence in the flow field. Fig. 17a and b display the convergence histories of versus time steps and CPU time respectively for both the explicit and implicit methods. The GMRES method is chosen as the linear solver in the implicit methods. For implicit DG (P1P2), the Jacobian matrix based on automatic differentiation (denoted as GMRES-AD-DG (P1P2)) provides the fastest convergence of the three methods. The one based on divided differencing (denoted as GMRES-DD-DG (P1P2)), although requires much fewer time steps than the one based on analytical

7.5. Laminar flow past a flat plate

The laminar boundary layer over an adiabatic flat plate at a free-stream Mach number of $M_\infty = 0.5$ and a Reynolds number of $Re = 100,000$ based on the free-stream velocity and the length of the flat plate is considered in this test case. This problem is chosen to assess (1) the accuracy of the WENO (P1P2) scheme for the discretization of the viscous and heat fluxes in the Navier–Stokes equations, as the classical Blasius solution can be used to measure the accuracy of the numerical solution, (2) performance of the implicit method for viscous flows. The computational domain is bounded from −0.5 to 1.0 along the x-direction, from 0 to 1.0 along the y-direction, and from 0 to 0.1 along the z-direction, and the no-slip surface starts at point (0.0, 0.2) and extends to (0.0, 0.2, 0.1). A slip condition is prescribed along the bottom side of the domain for x ∈ [−0.0, 0.0] with ν = 0. Symmetry conditions are prescribed for the front and back boundary with w = 0.
differentiation (denoted as MD-GMRES-DG (P1)), turns out to be the slowest in terms of CPU time among the three methods. For implicit WENO (P1P2), only the one based on AD (denoted as GMRES-AD-WENO (P1P2)) achieved an effective convergence in this test case, indicating the high robustness to use the AD technique to linearize the viscous terms in the WENO (P1P2) RDG method. Overall, the implicit methods are 30 times faster than their TVDK3 counterpart, demonstrating the superior advantage of using the developed implicit methods for viscous flow problems.

Secondly, computation is conducted on a prismatic + hexahedral hybrid grid (Npris = 977, Nhexa = 358) as shown in Fig. 18a. A zoom-in observation in Fig. 18b shows that the boundary layer region mainly consists of hexahedral elements, except at the leading edge region of the plate, where prismatic elements are filled. Fig. 18c shows the comparison of the computed $C_p$ distributions obtained by DG (P1) and WENO (P1P2) solutions with the analytical solution. A zoom-in observation for $x \in [0.02, 0.1]$ in Fig. 18d clearly demonstrates a more accurate prediction presented by WENO (P1P2) than DG (P1). Fig. 18e and f compare the computed velocity profiles with the analytical solutions at three downstream locations, where both the DG (P1) and WENO (P1P2) solutions again demonstrate a highly consistent convergence in the flow field.

Fig. 19a and b show the convergence histories versus time steps and CPU time respectively for the implicit DG (P1) and WENO (P1P2) methods (denoted as SGS (5)-AD-DG (P1) and SGS (5)-AD-WENO (P1P2)), in which the SGS (5) method is chosen as the linear

---

**Fig. 11.** Plot of pressure coefficient distributions at six span-wise locations for transonic flow over the ONERA M6 wing at $M_{\infty} = 0.84, \alpha = 3.06$: (a) $\eta = 0.20$, (b) $\eta = 0.44$, (c) $\eta = 0.65$, (d) $\eta = 0.80$, (e) $\eta = 0.90$, (f) $\eta = 0.95$. 

Y. Xia et al. / Computers & Fluids 98 (2014) 134–151

145
solver, and the Jacobian matrix is based on automatic differentiation. Compared with the implicit DG \((P_1)\), the implicit WENO \((P_1P_2)\) obtains the convergence without a significant increase in terms of both time steps and CPU time, demonstrating the order independence of the implicit method for computing viscous flows on hybrid grids.

---

**Fig. 12.** Convergence history versus (a) time steps and (b) CPU time for the implicit HWENO \((P_1P_2)\) method on 128 CPU processors, for transonic flow over the ONERA M6 wing at \(M_s = 0.84, \alpha = 3.06^\circ\).

**Fig. 13.** (a–b) The surface meshes of a tetrahedral grid. (c–d) Computed pressure contours on the unstructured surface mesh of a tetrahedral grid obtained by the HWENO \((P_1P_2)\) solutions for a transonic flow over a wing/pylon/finned-store configuration at \(M_s = 0.95, \alpha = 0^\circ\).

**Fig. 14.** Comparison of the computed pressure coefficient distributions with experimental data at two span-wise locations for a transonic flow over a wing/pylon/finned-store configuration at \(M_s = 0.95, \alpha = 0^\circ\): (a) \(\eta = 0.4077\), (b) \(\eta = 0.5900\).
Fig. 15. Convergence history versus (a) time steps and (b) CPU time for the implicit HWENO (P1P2) method on 128 CPU processors, for a transonic flow over a wing/pylon/finned-store configuration at $M_\infty = 0.95$, $\alpha = 0^\circ$.

Fig. 16. Plot of the tetrahedral grids for laminar flow past a flat plate at $Re = 100,000$: (a) global domain; (b) boundary layer region. (c) Logarithmic plot of the computed skin friction coefficient $c_f$ distribution along the flat plate $x \in [0, 1]$. (d) $c_f$ distribution at $x \in [0.5, 0.6]$. (e) $x$-velocity $u$ versus $\eta$. (f) Scaled $y$-velocity $v^+$ versus $\eta$. 

SGS5-HWENO(P1P2) on 128 CPUs

SGS5-HWENO(P1P2) on 128 CPUs
Fig. 17. Convergence histories versus (a) time steps and (b) CPU time using the explicit and implicit methods on the tetrahedral grid, for laminar flow past a flat plate at $Re = 100,000$.

Fig. 18. Plot of the prismatic + hexahedral hybrid grids for laminar flow past a flat plate at $Re = 100,000$: (a) global domain; (b) boundary layer region; (c) logarithmic plot of the computed skin friction coefficient $c_f$ distribution along the flat plate $x \in [0, 1]$; (d) $c_f$ distribution at $x = 0.02, 0.1$; (e) $x$-velocity $u$ versus $\eta$; (f) scaled $y$-velocity $v^+$ versus $\eta$. 
Fig. 19. Convergence histories versus (a) time steps and (b) CPU time using the implicit methods on prismatic + hexahedral hybrid grid, for laminar flow past a flat plate at $Re = 100,000$.

Fig. 20. A tetrahedral grid for subsonic flow over a sharp-edged slender delta wing at $M_\infty = 0.3, \alpha = 20.5^\circ$ and $Re = 0.95 \times 10^6$: (a) triangular surface meshes; (b) extracted meshes on sliced plane of $x/L = 0.9$.

Fig. 21. Surface pressure coefficient distributions obtained by the WENO ($P_1P_2$) solutions at four typical downstream intersections for subsonic flow over a delta wing at $M_\infty = 0.3, \alpha = 20.5^\circ$ and $Re = 0.95 \times 10^6$: (a) $x/L = 0.3$, (b) $x/L = 0.5$, (b) $x/L = 0.7$, (c) $x/L = 0.9$. 

Y. Xia et al. / Computers & Fluids 98 (2014) 134–151
7.6. Subsonic flow around a delta wing at Re $= 0.95 \times 10^6$

A subsonic flow around a sharp-edged slender delta wing at a free-stream Mach number of $M_a = 0.3$, angle of attack $\alpha = 20.5^\circ$, and Reynolds number of $Re = 0.95 \times 10^6$ based on a mean chord length of 1 is presented in this test case in order to assess the performance of our implicit WENO $(P_1P_2)$ RDG method for computing high Reynolds number flows, as the experimental data is available for the geometric configuration [68] considered. The aspect ratio of this delta wing is equal to 1, which corresponds to a sweep angle of $75.9638^\circ$. The upper surface of the wing is flat and the cross section is triangular ahead of $x/L = 0.9$, with the maximum thickness $0.021L$. The cross section downstream of $x/L = 0.9$ is trapezoidal, and the trailing edge is sharp. A tetrahedral grid ($Nellem = 316, 139$, $Npoin = 58, 322$, $Nafac = 21, 712$) is used in computation, as the triangular meshes of the wing surface are displayed in Fig. 20a, where the no-slip and adiabatic boundary conditions are prescribed. Fig. 20b displays a sliced-plane of $x/L = 0.9$, where one can see the highly stretched anisotropic elements piled on the wing surface. The height of the elements on the first layer is $0.5 \times 10^{-3}$, with a growth rate of 1.3 normal to the surface and 5 layers in total. The rest of the domain is filled with isotropic elements. Computation is initialized with constant free-stream values in the entire domain. The flow field is assumed to have reached the steady state using the implicit WENO $(P_1P_2)$ method (Jacobian matrix based on automatic differentiation) after a decrease of 8 orders of magnitude for the global density residual in only 191 time steps as shown in Fig. 22a. The convergence history versus CPU time is shown in Fig. 22b. Fig. 21a–d show the computed surface pressure coefficient distributions compared with the experimental data at four stations along the chord of the geometry. At the forward and middle stations as shown in Fig. 21a–c, the numerical results agree well with the experimental data, however the pressure coefficients are significantly over-predicted on the lower wing surface at station $x/L = 0.9$ as shown in Fig. 21d. Similar results were previously reported in Ref. [69,63]. The reason for the disagreement is not clearly known, but it was surmised that the disagreement might be caused by the presence of the pressure tubing that exits from the model in the lower surface trailing-edge region [69].

8. Conclusion

A set of implicit methods are developed for the third-order Hierarchical WENO $(P_1P_2)$ reconstruction based discontinuous Galerkin method for the solution of compressible flows on 3D hybrid grids. These implicit methods compute the Jacobian matrix arising from Newton linearization based on the underlying $P_1$ element approximation. In particular, three approaches: analytical derivation, divided differencing, and automatic differentiation are designed to construct the Jacobian matrix respectively, where the AD-based approach have shown the best robustness. Furthermore, these implicit methods are verified to compute the flows with discontinuities without any difficulty. The advantages of the developed implicit method lie in two aspects. On one side, this method can achieve the third-order accuracy, adding one order to the underlying implicit DG $(P_1)$ method without significant extra cost in computing time and memory requirement. On the other, this method only requires less than 1/6 of the memory space of the implicit DG $(P_2)$ method to achieve the same order of accuracy, resulting in a fast as well as low storage method that can be efficiently used to accelerate the convergence. An SPMD (single program, multiple data) programming paradigm based on MPI has been employed to achieve parallelism. The numerical results indicate that the developed implicit methods are orders of magnitude faster than their explicit counterpart. The performance of parallel computing shows that using these implicit methods, the HWENO $(P_1P_2)$ RDG method provides a viable and attractive DG solution for complicated flows of practical importance.

References


